Conservativity between logics and typed λ calculi*

Herman Geuvers**

Faculty of Mathematics and Computer Science
University of Nijmegen, Netherlands

1. Introduction

When looking at systems of typed λ calculus from a logical point of view, there are some interesting questions that arise. One of them is whether the formulas-as-types embedding from the logic into the typed λ calculus is complete, that is, whether the types that are inhabited in the typed λ calculus are provable (as formulas) in the logic. It is well-known that this is not a vacuous question: the 'standard' formulas-as-types embedding from higher order predicate logic into the Calculus of Constructions is not complete. (See [Berardi 1989], [Geuvers 1989] or [Geuvers 1993].) Another interesting issue is whether the typed λ calculus approach can help to solve questions about the logics or vice versa. An example of such a fruitful interaction is the proof of (strong) normalization for the Calculus of Constructions, which has as corollary in higher order predicate logic that cut elimination terminates. In this paper we want to treat questions of conservativity between systems of typed λ calculi (and hence between the logical systems that correspond with them according to the formulas-as-types embedding). On the one hand this is an issue of interest for the typed λ calculi themselves. (Can new type forming operators create inhabitants of previously empty types?) On the other hand, however, this is a nice example of how the formulas-as-types embedding can help to solve questions about logics by making use of typed λ calculi and vice versa.

If one sees a typed λ calculus as a logical system, one takes one specific universe ('sort' in the terminology of Pure Type Systems) to be interpreted as the universe of all formulas. Let's call this universe **Prop**. Now suppose that S_1 is a system of typed λ calculus containing the universe **Prop**, and suppose that S_2 is a system that extends S_1.

1.1. DEFINITION. The type system S_2 is a *conservative extension* of S_1 if for every context Γ and type A one has

$$\left.\begin{array}{l}\Gamma \vdash_{S_1} A : \mathsf{Prop} \\ \Gamma \vdash_{S_2} M : A\end{array}\right\} \Rightarrow \exists N[\Gamma \vdash_{S_1} N : A].$$

* Partially sponsored by the ESPRIT Basic Research Action "TYPES"
** e-mail: herman@cs.kun.nl

The 'logical' intuition should be clear: if the formula A, taken from the smaller system, is provable in the larger system, then it is already provable in the smaller system. To make the connection with logics a bit more precise we recall that, if L_1 and L_2 are logics and L_2 extends L_1, then L_2 is a *conservative extension* of L_1 if for all formulas φ and sets of fomulas Δ one has

$$\left.\begin{array}{r}\Delta \cup \{\varphi\} \text{ is a set of formulas of } L_1 \\ \Delta \vdash_{L_2} \varphi\end{array}\right\} \Rightarrow \Delta \vdash_{L_1} \varphi.$$

Now, let H be the formulas-as-types embedding from L_1 into S_1 and from L_2 into S_2. This means that for every finite set of formulas Δ' in L_i there is a specific context $\Gamma_{\Delta'}$ in S_i, in which all the declarations are made that are necessary for forming the types $H(\psi)$ (for $\psi \in \Delta$) in S_i. Furthermore this embedding H is *sound*:

$$\Delta \vdash_{L_i} \varphi \Rightarrow \exists N[\Gamma_{\Delta\cup\{\varphi\}}, \mathbf{p}{:}H(\Delta) \vdash_{S_i} N : H(\varphi)].$$

Here the $\mathbf{p}{:}H(\Delta)$ denotes a vector of variable-declarations $p_i : H(\psi_i)$ for each $\psi_i \in \Delta$. Hence the Δ is taken to be finite, which is not a real restriction. In the formulas-as-types embedding, the term N of type $H(\varphi)$ is defined by induction on the derivation of $\Delta \vdash \varphi$, so the formulas-as-types embedding not only maps formulas to types, but also derivations to proof-terms.

The formulas-as-types embedding is not always *complete*, where completeness means (in the terminology above) that for each formula φ and finite set of formulas Δ, taken from L_i one has

$$\Gamma_{\Delta\cup\{\varphi\}}, \mathbf{p}{:}H(\Delta) \vdash_{S_i} N : H(\varphi) \Rightarrow \Delta \vdash_{L_i} \varphi.$$

Some well-known examples of the formulas-as-types embedding are not complete, like the embedding of higher order predicate logic into the Calculus of Constructions. In this paper we are more interested in embeddings that are complete, in which case we usually speak of a *formulas-as-types isomorphism*. This is because of the following.

1.2. PROPOSITION. If the formulas-as-types embedding H is complete, then

$$S_2 \text{ is a conservative extension of } S_1 \Leftrightarrow L_2 \text{ is a conservative extension of } L_1.$$

This proposition can be useful in two ways, both of which will be applied in this paper. Note therefore that in the definition of conservativity (Definition 1.1) there is no requirement about a function that takes an inhabitant $M : \varphi$ in the larger system and returns an inhabitant $N : \varphi$ in the smaller system. However, if there is such a function, then the conservativity result will usually be much easier to prove for the typed λ calculi, because one just has to define the function and to show (by induction on the derivation or by induction on the structure of the term) that it preserves derivability. This is a purely *syntactic* conservativity proof. If it is not clear how such a function should be defined, it is better to look at the logics, in which case one can forget about the proof terms all together and just look at provability. In this latter case a semantic approach suits very well to prove conservativity.

Here we study the conservativity relations inside the cube of typed λ calculi, a collection of eight type systems defined by Barendregt (see [Barendregt 1992]) to give a fine structure for the Calculus of Constructions. There is a close connection between the cube of typed λ calculi and a cube of logical systems, due to the formulas-as-types embedding from the latter into the first. To make this embedding more readily understandable it is often described in two steps, first from the logic to a typed λ calculus that is in direct correspondence with the logic and then from this latter typed λ calculus to a type system of the cube. (See [Barendregt 1992] but also [Geuvers 1993].) To strip our discussions about typed λ calculi from the need to first having to justify all kinds of meta theoretic reasoning, we work in the framework of 'Pure Type Systems'. (See [Geuvers and Nederhof 1991], [Barendregt 1992] or [Geuvers 1993].) This gives a general method for describing typed λ calculi. Moreover we can use all the well-known meta-theory for Pure Type Systems (PTSs).

The main result in this paper is that, if S_2 is a system in the cube that contains the system S_1, then S_2 is a conservative extension of S_1, unless S_2 is the Calculus of Constructions (CC) and S_1 is the second order dependent typed λ calculus λP2. But more interesting than this general result is maybe the proof, which is divided into four cases. The first case is to show that CC is not conservative over λP2. The second case is to show that the extension of a system by adding type dependency is conservative. The third is to show that an extension of a first order system (i.e. a system in the bottom plane of the cube) is always conservative. This leaves over one special case, which is to show that $\lambda\omega$ is conservative over the polymorphic λ calculus, $\lambda 2$. The second and third case are dealt with by defining a mapping from terms in the larger system to the terms in the smaller system, which gives us a purely syntactic conservativity proof. The fourth case is more difficult, because it is not clear how to do this proof by purely syntactic means. We therefore define a semantics for the logical systems of higher and second order propositional logic (which are isomorphic to $\lambda\omega$ and $\lambda 2$ by the formulas-as-types embedding) and show the conservativity on the level of the logics.

2. A fine structure for the Calculus of Constructions

2.1. Pure Type Systems

Our studies of the Calculus of Constructions and its subsystems will be done in the framework of 'Pure Type Systems'. This provides a generic way of describing systems of typed λ calculus. In fact one can only describe systems that have as type forming operator just the Π and as reduction rule just β. As the Calculus of Constructions is such a system, the Pure Type Systems (or PTSs) is the right framework for us.

The Pure Type Systems are formal systems for deriving judgements of the form

$$\Gamma \vdash M : A,$$

where both M and A are in the set of so called *pseudoterms*, a set of expressions from which the derivation rules select the ones that are typable. The Γ is a finite sequence of *declarations*, statements of the form $x : B$, where x is a variable and B is a pseudoterm. The idea is that a term M can only be of type A (notation $M : A$) relative to a typing of the free variables that occur in M and A. Before giving the precise definition of Pure Type Systems we define the set of pseudoterms T over a base set $\mathcal{S}$. (The dependency of T on $\mathcal{S}$ is usually ignored.)

2.1. DEFINITION. For $\mathcal{S}$ some set, the set of pseudoterms over $\mathcal{S}$, T, is defined by

$$\mathsf{T} ::= \mathcal{S} \mid \mathsf{Var} \mid (\Pi \mathsf{Var}{:}\mathsf{T}.\mathsf{T}) \mid (\lambda \mathsf{Var}{:}\mathsf{T}.\mathsf{T}) \mid \mathsf{TT},$$

where Var is a countable set of expressions, called variables. Both Π and λ bind variables and hence we have the usual notions of *free variable* and *bound variable*. We adopt the λ-calculus notation of writing $\mathrm{FV}(M)$ for the set of free variables in the pseudoterm M.
On T we have the usual notion of β-reduction, generated from

$$(\lambda x{:}A.M)P \longrightarrow_\beta M[P/x],$$

where $M[P/x]$ denotes the substitution of P for x in M (done with the usual care to avoid capturing of free variables), and compatible with application, λ-abstraction and Π-abstraction. We also adopt from the untyped λ calculus the conventions of denoting the transitive reflexive closure of $\longrightarrow_\beta$ by $\longrightarrow\!\!\!\!\!\rightarrow_\beta$ and the transitive symmetric closure of $\longrightarrow\!\!\!\!\!\rightarrow_\beta$ by $=_\beta$.

The typing of terms is done under the assumption of specific types for the free variables that occur in the term.

2.2. DEFINITION. 1. A *declaration* is a statement of the form $x : A$, where x is a variable and A a pseudoterm,

2. A *pseudocontext* is a finite sequence of declarations such that, if $x : A$ and $y : B$ are different declarations of the same pseudocontext, then $x \not\equiv y$,
3. If $\Gamma = x_1{:}A_1, \ldots, x_n{:}A_n$ is a pseudocontext, the *domain of* Γ, $\mathsf{dom}(\Gamma)$ is the set $\{x_1, \ldots, x_n\}$; for $x_i \in \mathsf{dom}(\Gamma)$.
4. For Γ a pseudocontext, a variable y is *Γ-fresh* (or just *fresh* if it is clear which Γ we are talking about) if $y \notin \mathsf{dom}(\Gamma)$.

2.3. DEFINITION. A *Pure Type System* (PTS) is given by a set $\mathcal{S}$, a set $\mathcal{A} \subset \mathcal{S} \times \mathcal{S}$ and a set $\mathcal{R} \subset \mathcal{S} \times \mathcal{S} \times \mathcal{S}$. The PTS that is given by $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ is denoted by

$\lambda(\mathcal{S},\mathcal{A},\mathcal{R})$ and is the typed λ calculus with the following deduction rules.

$$\text{(sort)}\quad \vdash s_1 : s_2 \qquad \text{if } (s_1,s_2)\in\mathcal{A}$$

$$\text{(var)}\quad \frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A} \qquad \text{if } x \text{ is } \Gamma\text{-fresh}$$

$$\text{(weak)}\quad \frac{\Gamma \vdash A : s \quad \Gamma \vdash M : C}{\Gamma, x{:}A \vdash M : C} \qquad \text{if } x \text{ is } \Gamma\text{-fresh}$$

$$(\Pi)\quad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_3} \qquad \text{if } (s_1,s_2,s_3)\in\mathcal{R}$$

$$(\lambda)\quad \frac{\Gamma, x{:}A \vdash M : B \quad \Gamma \vdash \Pi x{:}A.B : s}{\Gamma \vdash \lambda x{:}A.M : \Pi x{:}A.B}$$

$$\text{(app)}\quad \frac{\Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$$

$$\text{(conv)}\quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B} \qquad A =_\beta B$$

If $s_2 \equiv s_3$ in a triple $(s_1,s_2,s_3)\in\mathcal{R}$, we write $(s_1,s_2)\in\mathcal{R}$. The equality in the conversion rule (conv) is the β-equality on the set of pseudoterms T.
The elements of $\mathcal{S}$ are called *sorts*, the elements of $\mathcal{A}$ (usually written as $s_1 : s_2$) are called *axioms* and the elements of $\mathcal{R}$ are called *rules*.
If $\lambda(\mathcal{S},\mathcal{A},\mathcal{R})$ is a PTS, the set of *terms* of $\lambda(\mathcal{S},\mathcal{A},\mathcal{R})$, $\mathsf{Term}(\lambda(\mathcal{S},\mathcal{A},\mathcal{R}))$, is defined by

$$\mathsf{Term}(\lambda(\mathcal{S},\mathcal{A},\mathcal{R})) = \{A \mid \exists \Gamma, B[\Gamma \vdash A : B \vee \Gamma \vdash B : A]\}.$$

This is not the place to go into a detailed treatment of the meta-theoretic properties of PTSs. We refer to [Geuvers and Nederhof 1991], [Barendregt 1992] or [Geuvers 1993] for details. We only give the most important properties without proof.

2.4. PROPOSITION. In an arbitrary PTS $\lambda(\mathcal{S},\mathcal{A},\mathcal{R})$, the following holds.

- Substitution
  If $\Gamma_1, x{:}A, \Gamma_2 \vdash M : B$ and $\Gamma_1 \vdash N : A$, then $\Gamma_1, \Gamma_2[N/x] \vdash M[N/x] : B[N/x]$.
- Stripping

(i) $\Gamma \vdash s : R,\ s \in \mathcal{S} \Rightarrow R =_\beta s'$ with $s : s' \in \mathcal{A}$ for some $s' \in \mathcal{S}$,
(ii) $\Gamma \vdash x : R,\ x \in \mathsf{Var} \Rightarrow R =_\beta A$ with $x : A \in \Gamma$ for some term A,
(iii) $\Gamma \vdash \Pi x{:}A.B : R \Rightarrow \Gamma \vdash A : s_1, \Gamma, x{:}A \vdash B : s_2$ and $R =_\beta s_3$ with $(s_1, s_2, s_3) \in \mathcal{R}$ for some $s_1, s_2, s_3 \in \mathcal{S}$,
(iv) $\Gamma \vdash \lambda x{:}A.M : R \Rightarrow \Gamma, x{:}A \vdash M : B, \Gamma \vdash \Pi x{:}A.B : s$ and $R =_\beta \Pi x{:}A.B$ for some term B and $s \in \mathcal{S}$,
(v) $\Gamma \vdash MN : R \Rightarrow \Gamma \vdash M : \Pi x{:}A.B, \Gamma \vdash N : A$ with $R =_\beta B[N/x]$ for some terms A and B.

- Subject Reduction
  If $\Gamma \vdash M : A$ and $M \longrightarrow\!\!\!\!\!\rightarrow_\beta N$, then $\Gamma \vdash N : A$.
- Confluence
  If $\Gamma \vdash M : A$, $\Gamma \vdash N : A$ and $M =_\beta N$, then there is a term Q with $\Gamma \vdash Q : A$ and $M \longrightarrow\!\!\!\!\!\rightarrow_\beta Q$, $N \longrightarrow\!\!\!\!\!\rightarrow_\beta Q$.

The definition of Pure Type System gives rise to an interesting notion of morphism between typed λ calculi which can be described by taking into account only the sorts, axioms and rules of the system.

2.5. DEFINITION. Let $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ and $\lambda(\mathcal{S}', \mathcal{A}', \mathcal{R}')$ be PTSs. A *morphism from* $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ *to* $\lambda(\mathcal{S}', \mathcal{A}', \mathcal{R}')$ is a mapping f from $\mathcal{S}$ to $\mathcal{S}'$ that *preserves axioms and rules*, that is

$$s_1{:}s_2 \in \mathcal{S} \Rightarrow f(s_1){:}f(s_2) \in \mathcal{S}',$$
$$(s_1, s_2, s_3) \in \mathcal{R} \Rightarrow (f(s_1), f(s_2), f(s_3)) \in \mathcal{R}'.$$

A PTS-morphism f from $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ to $\lambda(\mathcal{S}', \mathcal{A}', \mathcal{R}')$ immediately extends to a mapping from the pseudoterms of $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ to the pseudoterms of $\lambda(\mathcal{S}', \mathcal{A}', \mathcal{R}')$ and hence to a mapping from pseudocontexts to pseudocontexts. This mapping preserves substitution and β-equality and also derivability:

2.6. LEMMA. If f is a PTS-morphism from ζ to ζ', then

$$\Gamma \vdash_\zeta M : A \Rightarrow f(\Gamma) \vdash_{\zeta'} f(M) : f(A).$$

2.2. The cube of typed λ calculi

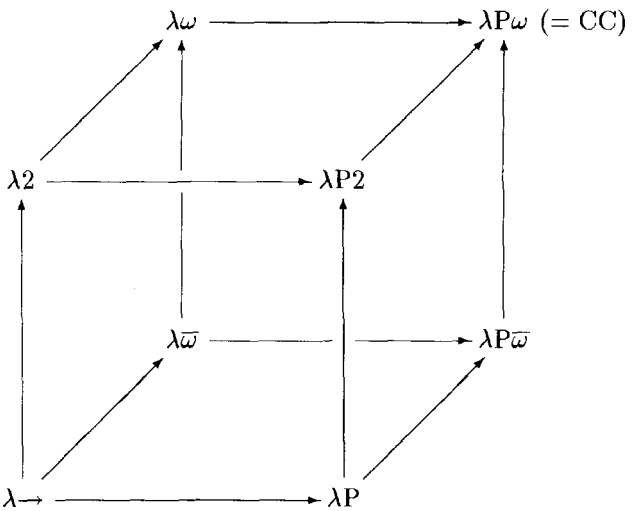
2.7. DEFINITION. The Barendregt's cube of typed λ calculi consists of eight PTSs. Each of them has

$$\mathcal{S} := \{\star, \Box\},$$
$$\mathcal{A} := \{\star : \Box\}.$$

The set of rules $\mathcal{R}$ for each system are as given in the following table.

$\lambda\!\!\rightarrow$	$(\star,\star)$			
$\lambda 2$	$(\star,\star)$	$(\Box,\star)$		
$\lambda\overline{\omega}$	$(\star,\star)$		$(\Box,\Box)$	
$\lambda\omega$	$(\star,\star)$	$(\Box,\star)$	$(\Box,\Box)$	
λP	$(\star,\star)$	$(\star,\Box)$		
$\lambda P2$	$(\star,\star)$	$(\star,\Box)$	$(\Box,\star)$	
$\lambda P\overline{\omega}$	$(\star,\star)$	$(\star,\Box)$	$(\Box,\Box)$	
$\lambda P\omega$	$(\star,\star)$	$(\star,\Box)$	$(\Box,\star)$	$(\Box,\Box)$.

The system $\lambda P\omega$ is the Calculus of Constructions, sometimes called the *Pure* Calculus of Constructions to distinguish it from its variants and extensions. We refer to it as CC. The systems of the cube are usually presented as follows.



where an arrow denotes inclusion of one system in another.

The systems $\lambda\!\!\rightarrow$ and $\lambda 2$ are also known as the simply typed lambda calculus and the polymorphically typed λ calculus (due to Girard, as system F, and Reynolds). The system $\lambda\omega$ is a higher order version of $\lambda 2$, also known as Girard's system Fω. The presentation of these systems as a PTS is quite different from the original one. If one is just interested in those systems alone it is in general more convenient to study them in their original presentation. The PTS framework is more convenient for systems with *type dependency*, that is the feature that a type $A{:}\star$ may itself contain a subterm M with $M{:}B{:}\star$. This situation only occurs in the presence of the rule $(\star,\Box)$. In that case there is no other syntax for the systems which is essentially more convenient then the PTS format. The system λP is very close to the system LF [Harper et al. 1987]. In fact LF is obtained from λP by replacing in the conversion rule the side condition $A =_\beta B$ by $A =_{\beta\eta} B$. The system $\lambda P\omega$ is the Calculus of Constructions, due to [Coquand 1985]. (See

also [Coquand and Huet 1988].) The system $\lambda P2$ was defined under the same name in [Longo and Moggi 1988].

The formulas-as-types embedding from logical systems into the systems of the cube is best understood by first defining a cube of eight 'logical typed λ calculi'. These are systems for which there is a clear one-to-one correspondence between the original logical system and the typed λ calculus. This correspondence is given by the formulas-as-types embedding. This embedding assigns to every formula φ a type $\tilde{\varphi}$ and to every proof in natural deduction style a term such that a proof of φ becomes a term of the type $\tilde{\varphi}$. That this embedding is one-to-one means that every term of the type $\tilde{\varphi}$ is the image of a proof of φ.

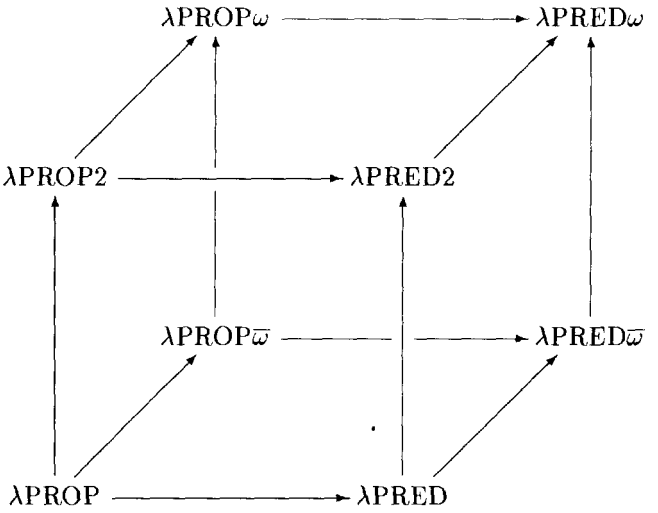
2.8. DEFINITION ([Berardi 1990]). The *logic cube* consists of eight PTSs, each of them having as sorts and axioms

$$\mathcal{S} = \mathsf{Prop}, \mathsf{Set}, \mathsf{Type}^p, \mathsf{Type}^s,$$
$$\mathcal{A} = \mathsf{Prop} : \mathsf{Type}^p, \mathsf{Set} : \mathsf{Type}^s.$$

The rules of each of the systems are given by the following table

λPROP				
	$(\mathsf{Prop}, \mathsf{Prop})$			
λPROP2				
	$(\mathsf{Prop}, \mathsf{Prop})$		$(\mathsf{Type}^p, \mathsf{Prop})$	
λPROP$\overline{\omega}$				$(\mathsf{Type}^p, \mathsf{Type}^p)$
	$(\mathsf{Prop}, \mathsf{Prop})$			
λPROPω				$(\mathsf{Type}^p, \mathsf{Type}^p)$
	$(\mathsf{Prop}, \mathsf{Prop})$		$(\mathsf{Type}^p, \mathsf{Prop})$	
λPRED	$(\mathsf{Set}, \mathsf{Set})$	$(\mathsf{Set}, \mathsf{Type}^p)$		
	$(\mathsf{Prop}, \mathsf{Prop})$	$(\mathsf{Set}, \mathsf{Prop})$		
λPRED2	$(\mathsf{Set}, \mathsf{Set})$	$(\mathsf{Set}, \mathsf{Type}^p)$		
	$(\mathsf{Prop}, \mathsf{Prop})$	$(\mathsf{Set}, \mathsf{Prop})$	$(\mathsf{Type}^p, \mathsf{Prop})$	
λPRED$\overline{\omega}$	$(\mathsf{Set}, \mathsf{Set})$	$(\mathsf{Set}, \mathsf{Type}^p)$	$(\mathsf{Type}^p, \mathsf{Set})$	$(\mathsf{Type}^p, \mathsf{Type}^p)$
	$(\mathsf{Prop}, \mathsf{Prop})$	$(\mathsf{Set}, \mathsf{Prop})$		
λPREDω	$(\mathsf{Set}, \mathsf{Set})$	$(\mathsf{Set}, \mathsf{Type}^p)$	$(\mathsf{Type}^p, \mathsf{Set})$	$(\mathsf{Type}^p, \mathsf{Type}^p)$
	$(\mathsf{Prop}, \mathsf{Prop})$	$(\mathsf{Set}, \mathsf{Prop})$	$(\mathsf{Type}^p, \mathsf{Prop})$	

The systems are presented in a picture as follows.



where an arrow denotes inclusion of one system in another.

That the type systems described above indeed correspond to logical systems will not be discussed here. See [Barendregt 1992], [Tonino and Fujita 1992] and [Geuvers 1993] for details. To get the idea we give some examples.

2.9. EXAMPLES. 1. A:Set, R:$A\to A\to$Prop, φ:Prop $\vdash$
$\lambda p{:}(\Pi x, y{:}A.Rxy\to Ryx\to\varphi).\lambda x{:}A.\lambda q{:}Rxx.pxxqq$:
$(\Pi x, y{:}A.Rxy\to Ryx\to\varphi)\to(\Pi x{:}A.Rxx\to\varphi)$ in λPRED.
The term $R : A\to A\to$Prop is understood as a binary relation on A. It should be clear how the term $\lambda x{:}A.\lambda q{:}Rxx.pxxqq$ corresponds to a proof in natural deduction style of the proposition
$(\forall x, y \in A[R(x,y) \supset R(y,x) \supset \varphi]) \supset (\forall x \in A[R(x,x)\to\varphi])$.

2. In any system that contains λPROP2, $\bot$ can be defined as $\Pi\alpha$:Prop.α(: Prop). One has indeed φ:Prop $\vdash \lambda p{:}\bot.p\varphi : \bot\to\varphi$.

3. In λPRED2 one has A:Set $\vdash$
$\lambda R{:}A\to A\to$Prop.$\lambda p{:}(\Pi x, y{:}A.Rxy\to Ryx\to\bot).\lambda x{:}A.\lambda q{:}Rxx.pxxqq$:
$\Pi R{:}A\to A\to$Prop.$(\Pi x, y{:}A.Rxy\to Ryx\to\bot)\to(\Pi x{:}A.Rxx\to\bot)$, stating that any binary relation that is antisymmetric is areflexive.

The systems of Barendregt's cube and the logic cube enjoy some more special properties that will be used. First of all, the type of a term is unique up to β-conversion. (The proof is by induction on terms, see [Geuvers 1993] or [Barendregt 1992].)

2.10. PROPOSITION (Uniqueness of Types). For a system in one of the two cubes one has that if $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$, then $A =_\beta B$.

Another nice thing is that, if variables are treated with some care, then the terms can be classified into disjoint sets. One therefore divides the set of variables Var into disjoint subsets Var^s $(s \in \mathcal{S})$. In the rules (weak) and (var), if $\Gamma \vdash A : s$ is the premise, one can now only take a variable x from the set Var^s. In the type systems of Barendregt's cube, one often uses Greek characters and capitals for the variables in $\mathsf{Var}^\Box$ and latin characters for the variables in $\mathsf{Var}^\star$. The following definition is now useful.

2.11. DEFINITION. Let us consider any system of Barendregt's cube.

1. The set of kinds is defined by $\mathsf{Kind} := \{A \mid \exists \Gamma[\Gamma \vdash A : \Box]\}$.
2. The set of types is defined by $\mathsf{Type} := \{A \mid \exists \Gamma[\Gamma \vdash A : \star]\}$.
3. The set of constructors is defined by $\mathsf{Constr} := \{P \mid \exists A, \Gamma[\Gamma \vdash P : A : \Box]\}$.
4. The set of objects is defined by $\mathsf{Obj} := \{P \mid \exists A, \Gamma[\Gamma \vdash P : A : \star]\}$.

Here $\Gamma \vdash P : A : \star$ denotes the fact that $\Gamma \vdash P : A$ and $\Gamma \vdash A : \star$.

The usefulness of this definition is due to the following lemma. (For a detailed proof see [Geuvers 1993].)

2.12. LEMMA (Classification). Let us consider any system of Barendregt's cube.

$$\mathsf{Kind} \cap \mathsf{Type} = \emptyset,$$
$$\mathsf{Constr} \cap \mathsf{Obj} = \emptyset.$$

The formulas-as-types embedding of a logic into the corresponding system of the logic cube is an isomorphism (for details see [Geuvers 1993]), so we can restrict our study of the formulas-as-types embedding into the systems of Barendregt's cube to the study of the *collapsing mapping* H that maps the systems of the logic cube into the ones of the cube of typed λ calculi.

2.13. DEFINITION. The collapsing mapping H is defined as the family of PTS-morphisms from logic cube to Barendregt's cube given by

$$H(\mathsf{Prop}) = \star,$$
$$H(\mathsf{Set}) = \star,$$
$$H(\mathsf{Type}^p) = \Box,$$
$$H(\mathsf{Type}^s) = \Box.$$

It is immediate that the collapsing mapping H does not really do anything for the systems of the left plane of the cube. The sorts Prop and Type^p are renamed as $\star$ and $\Box$, but there are no additional rules. This implies that the formulas-as-types embedding from propositional logics into a system of the left plane of the cube is an isomorphism. For the right plane of the cube the situation is more interesting, because the sorts Prop and Set, respectively Type^p and Type^s are mapped to the same sort in the Barendregt's cube. The question of *completeness* of H becomes a real issue here.

2.14. DEFINITION. For L_i a system of the logic cube and S_i the corresponding system in Barendregt's cube, we say that $H : L_i \to S_i$ is *complete* if for all contexts Γ in L_i and φ with $\Gamma \vdash_{L_i} \varphi : \mathsf{Prop}$, one has

$$H(\Gamma) \vdash_{S_i} M : H(\varphi) \Rightarrow \exists N[\Gamma \vdash_{L_i} N : \varphi].$$

Completeness is of course important because typed λ calculi like the Calculus of Constructions are intended to be used as systems for formalizing mathematics, which is done by reasoning in the embedded higher order predicate logic. One then tacitly assumes that completeness holds, at least for the specific set of formulas that one is interested in.

However, the mapping H identifies the universe of sets (Set) and the universe of propositions (Prop), which is a strange feature in a logical system. For example, it implies that an axiom that states a property for all propositions, can also be applied to sets after the embeding H. In the higher order case (third order and higher) this can be used to construct counterexamples to completeness. A short counterexample, due to [Geuvers 1989] is found by considering the proposition $\Pi x{:}A.\phi$, with $x \notin \mathrm{FV}(\phi)$, and assuming $Q(\Pi x{:}A.\phi)$ for a basic predicate Q : Prop$\to$Prop. Then the proposition

$$\exists \psi{:}\mathsf{Prop}.Q(\psi \to \phi)$$

is not provable in λPREDω, whereas in CC, a term of type $\exists \psi{:}\star.Q(\psi \to \phi)$ can easily be found. (Take for ψ just $A : \star$.) Another counterexample, which is not so much of purely syntactical nature, is due to [Berardi 1989]. It is found by considering the extensionality axiom

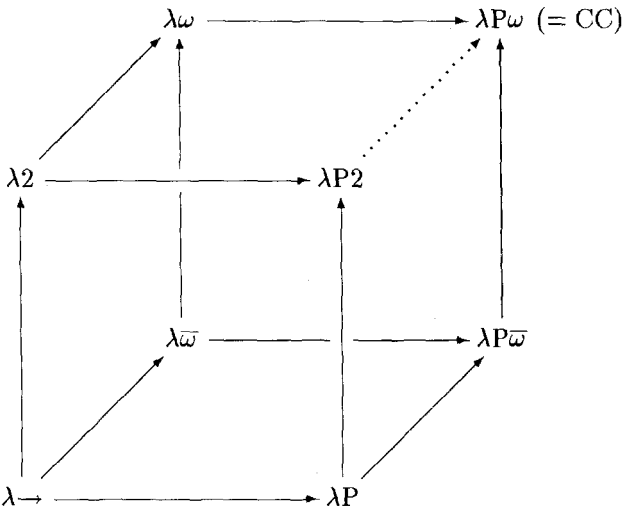
$$\mathrm{ext} := \Pi \phi, \psi{:}\mathsf{Prop}.(\phi \leftrightarrow \psi) \to (\phi = \psi).$$

Let Γ be the context $A : \mathsf{Set}, a, a' : A, z : a \neq a', y : \mathrm{ext}$. (Here, the equality denotes the so called 'Leibniz' equality, defined by taking for $t, u : B$, $t = u$ to be $\Pi P : B \to \mathsf{Prop}.Pt \to Pu$. This equivalence relation identifies terms that have the same properties.) Now, if one looks at the behaviour of ext in CC after the embedding H, it can be observed that $H(\mathrm{ext})$ also applies to A. Hence, in CC we can derive in the context $H(\Gamma)$ that $A = A \to A$ holds. But this implies that A is a λ-algebra (this statement can be formalised precisely), which is of course not the case in λPREDω. More details about incompleteness of $H : \lambda\mathrm{PRED}\omega \to \mathrm{CC}$ can be found in [Barendregt 1992] and [Geuvers 1993].

The two counterexamples to completeness of H that are discussed above, apply to nth order predicate logic for any $n > 2$. Hence the embedding $H : \lambda\mathrm{PRED}n \to \lambda Pn$ is incomplete for any $n > 2$. (These systems λPREDn and λPn are found by a straightforward extension of the second order systems, allowing quantification over the collection of domains of order $\leq n$.) In contrast, the embedding $H : \lambda\mathrm{PRED} \to \lambda\mathrm{P}$ is complete, as was proved in [Berardi 1989] and [Barendsen and Geuvers 1989]. For the embedding $H : \lambda\mathrm{PRED2} \to \lambda\mathrm{P2}$ the question of completeness is still open.

3. Conservativity relations inside the cube

We now want to address the question of conservativity inside the cube of typed λ calculi and the logic cube. We first look at the cube of typed λ calculi, because the situation for the logic cube is very similar. There are four results that do the whole job, resulting in the following picture.



where an arrow denotes a conservative inclusion and a dotted arrow denotes a non-conservative inclusion. By transitivity of conservativity (if system 3 is conservative over system 2 and system 2 is conservative over system 1, then system 3 is conservative over system 1), it is no problem to fill in the picture further. (Draw the arrows between two non-adjacent systems) We can collect all this in the following Proposition.

3.1. THEOREM. For S_1 and S_2 two systems in the cube of typed lambda calculi such that $S_1 \subseteq S_2$:

$$S_2 \text{ is conservative over } S_1 \Leftrightarrow S_2 \neq \text{CC} \vee S_1 \neq \lambda\text{P2}.$$

PROOF. It suffices to prove the following four results.

1. If $S_2 \supseteq S_1$, with S_1 a system of the lower plane in the cube, then S_2 is conservative over S_1.(Proposition 3.2.)
2. If S_2 is a system in the right plane of the cube, and S_1 is the adjacent system in the left plane, then S_2 is conservative over S_1.(Proposition 3.6.)
3. $\lambda\text{P}\omega$ is not conservative over λP2,
4. $\lambda\omega$ is conservative over λ2. (Corollary 4.20.)

The fourth is a consequence of Corollary 4.20, saying that PROPω is conservative over PROP2, and of the fact that PROPω and PROP2 are isomorphic to, respectively, $\lambda\omega$ and λ2 via the formulas-as-types embedding. The conservativity of PROPω over PROP2 will be proved in detail later by using semantical methods.

The third was verified in detail by [Ruys 1991], following an idea from Berardi. The idea is to look at a context Γ in λP2 that represents Arithmetic. Then Γ with λP2 is as strong as second order Arithmetic and Γ with $\lambda P\omega$ is as strong as higher order Arithmetic. Hence we can use Gödel's Second Incompleteness Theorem to show that in λP2 one can not derive from Γ that Γ is consistent in λP2. On the other hand in $\lambda P\omega$ one can derive from Γ that Γ is consistent in λP2. Hence the non-conservativity. □

We first prove the Proposition about conservativity of systems over systems in the lower plane. The Proposition was also proved in [Verschuren 1990] in a slightly different way.

3.2. PROPOSITION. We consider the cube of typed λ calculi. Let S_1 be a system of the lower plane and S_2 be another system of the cube such that $S_1 \subseteq S_2$. Then

$$\left.\begin{array}{r} \Gamma \vdash_{S_1} B : \star \\ \Gamma \vdash_{S_2} M : B \\ \Gamma \text{ and } M \text{ in normal form} \end{array}\right\} \Rightarrow \Gamma \vdash_{S_1} M : B.$$

PROOF. By induction on the structure of M.

var. Say $M \equiv x$. Then $x : A \in \Gamma$, with $A =_\beta B$. Hence, $\Gamma \vdash_{S_1} x : A$, and by one application of (conv), we can conclude that $\Gamma \vdash_{S_1} x : B$.

applic. Say $M \equiv xP_1 \cdots P_n$. Then, by Stripping, $x{:}A \in \Gamma$ with $A =_\beta \Pi y_1{:}C.D$ for some C and D. Now, A is in normal form (because Γ is) and so A is itself a Π-term, say $A \equiv \Pi y_1{:}C_1.D_1$. So, $x{:}\Pi y_1{:}C_1.D_1 \in \Gamma$. Now, $\Gamma \vdash_{S_1} \Pi y_1{:}C_1.D_1 : \star$ (in the lower plane, i.e. without the rule $(\Box, \star)$), but then also

$$\Gamma \vdash_{S_1} C_1 : \star.$$

Of course we also have

$$\Gamma \vdash_{S_2} P_1 : C_1,$$

so by IH (note that P_1 is in normal form), $\Gamma \vdash_{S_1} P_1 : C_1$. Hence $\Gamma \vdash_{S_1} xP_1 : D_1[P_1/y_1]$. We can now go further with P_2: We know that $D_1[P_1/y_1] \twoheadrightarrow_\beta \Pi y_2{:}C_2.D_2$. Now, $\Gamma \vdash_{S_1} D_1[P_1/y_1] : \star$ and hence $\Gamma \vdash_{S_1} \Pi y_2{:}C_2.D_2 : \star$ by Subject Reduction. So

$$\Gamma \vdash_{S_1} C_2 : \star.$$

Also

$$\Gamma \vdash_{S_2} P_2 : C_2,$$

so again we can apply IH to obtain $\Gamma \vdash_{S_1} P_2 : C_2$ and hence we have $\Gamma \vdash_{S_1} xP_1P_2 : D_2[P_2/y_2]$. Continuing in this way upto n we find that $\Gamma \vdash_{S_1} xP_1 \cdots P_n : D_n[P_n/y_n]$ with $D_n[P_n/y_n] =_\beta B$. By one application of conversion (using $\Gamma \vdash_{S_1} B : \star$) we conclude $\Gamma \vdash_{S_1} xP_1 \cdots P_n : B$.

abstr. Say $M \equiv \lambda x{:}A.N$. Then $B \twoheadrightarrow_\beta \Pi x{:}A.C$ for some C (note that A is in normal form). So $\Gamma \vdash_{S_1} \Pi x{:}A.C : \star$ by Subject Reduction and $\Gamma, x{:}A \vdash_{S_2} N : C$ (by Stripping and the conversion rule). By IH we conclude $\Gamma, x{:}A \vdash_{S_1} N : C$. Now we are done: By one λ-abstraction and one conversion we conclude $\Gamma \vdash_{S_1} \lambda x{:}A.N : B$. □

The side condition Γ in normal form has just been added for convenience (in giving the proof.) It is not essential and it may be dropped.

As a corollary one finds that a system of the cube is conservative over all its subsystems of the lower plane. If S_1 in the lower plane and $S_1 \subseteq S_2$, then

$$\left.\begin{array}{l}\Gamma \vdash_{S_1} A : \star \\ \Gamma \vdash_{S_2} M : A\end{array}\right\} \Rightarrow \exists N[\Gamma \vdash_{S_1} N : A].$$

This can even be made more precise, because the term N can be computed directly from the term M as follows.

3.3. COROLLARY. If $S_1 \subseteq S_2$, with S_1 in the lower plane, then

$$\left.\begin{array}{l}\Gamma \vdash_{S_1} A : \star \\ \Gamma \vdash_{S_2} M : A\end{array}\right\} \Rightarrow \Gamma \vdash_{S_1} \mathrm{nf}(M) : A,$$

where $\mathrm{nf}(M)$ denotes the β-normal form of M.

We now give a proof of the conservativity of the right plane over the left plane. The idea is to define a mapping that removes all type dependencies. This mapping will go from a system in the right plane to the adjacent system in the left plane and is the identity on terms that are already well-typed in the left plane. Hence the conservativity. The proof is originally independently due to [Paulin 1989] and [Berardi 1990]. The first described the mapping from $\lambda P\omega$ to $\lambda\omega$ in the first place to use it for program extraction; the second described the collection of four mappings (which is a straightforward generalisation of the mapping from $\lambda P\omega$ to $\lambda\omega$) to give this conservativity proof. The mappings are very much related to similar mappings one can define from predicate logic to propositional logic to prove conservativity of the first over the second.

3.4. DEFINITION ([Paulin 1989], [Berardi 1990]). Let S_2 be a system of the right plane and S_1 the adjacent system in the left plane. The mapping $[-] : \mathsf{Term}(S_2) \to \mathsf{Term}(S_1)$ is defined as follows.

$$\begin{aligned}
[\Box] &= \Box,\\
[\star] &= \star,\\
[x] &= x, \text{ for } x \text{ a variable},\\
[\Pi x{:}A.B] &= [B] \text{ if } A{:}\star, B{:}\Box,\\
&= \Pi x{:}[A].[B] \text{ else},\\
[\lambda x{:}A.M] &= [M] \text{ if } A{:}\star, M{:}B{:}\Box, \text{ (for some } B),\\
&= \lambda x{:}[A].[M] \text{ else},\\
[PM] &= [P] \text{ if } M{:}A{:}\star, P{:}B{:}\Box, \text{ (for some } A, B),\\
&= [P][M] \text{ else},
\end{aligned}$$

3.5. REMARK. The side conditions in the definition are justified by the Classification Lemma (2.12).

The mapping $[-]$ extends straightforwardly to contexts. The following proposition justifies the statement in the definition that the mapping $[-]$ goes from the right plane to the left plane.

3.6. PROPOSITION ([Paulin 1989], [Berardi 1990]). Let S_2 be a system in the right plane and S_1 the adjacent system in the left plane of the cube.

$$\Gamma \vdash_{S_2} M : A \Rightarrow [\Gamma] \vdash_{S_1} [M] : [A]$$

PROOF. By a straightforward induction on the derivation of $\Gamma \vdash_{S_2} M : A$. □

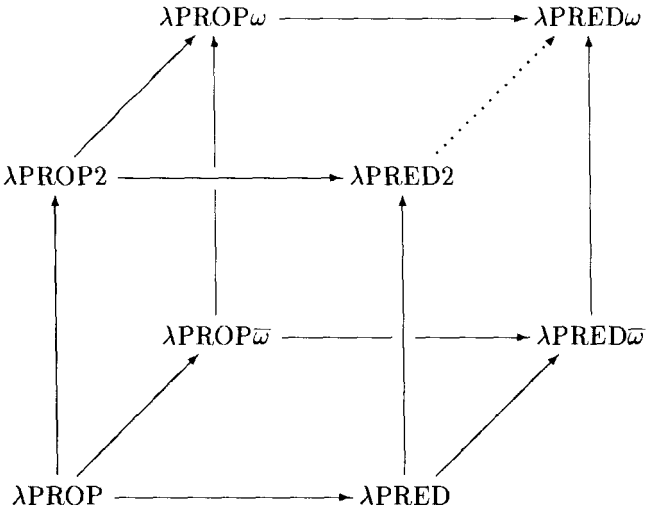
3.7. COROLLARY ([Paulin 1989],[Berardi 1990]). For S_2 a system in the right plane and S_1 the adjacent system in the left plane of the cube we have that S_2 is conservative over S_1.

PROOF. The only thing to check is that for $M \in \mathsf{Term}(S_1)$, $[M] \equiv M$. This is done by an easy induction on the structure of M. □

Corollary 3.7 can be made a bit more precise by stating how the term in the smaller system is computed from the term in the larger system. For S_2 in the right plane and S_1 the adjacent system in the left plane one has

$$\left.\begin{array}{l}\Gamma \vdash_{S_1} A : \star \\ \Gamma \vdash_{S_2} M : A\end{array}\right\} \Rightarrow \Gamma \vdash_{S_1} [M] : A,$$

The conservativity relations in the logic cube (Definition 2.8) are as follows. (An arrow denotes a conservative extension, a dotted arrow a non-conservative extension.)



3.8. PROPOSITION. For L_1 and L_2 two systems in the logic cube such that $L_1 \subseteq L_2$:

$$L_2 \text{ is conservative over } L_1 \Leftrightarrow L_2 \neq \lambda\text{PRED}\omega \vee L_1 \neq \lambda\text{PRED2}.$$

PROOF. The proof is completely analoguous to the proof for the cube of typed lambda calculi. What has to be proved for conservativity is that, if $L_1 \subset L_2$ and Γ is a context of L_1 with $\Gamma \vdash A : \mathsf{Prop}$, then

$$\Gamma \vdash_{L_2} M : A \Rightarrow \exists N[\Gamma \vdash_{L_1} N : A].$$

The proof of non-conservativity of $\lambda\text{PRED}\omega$ over λPRED2 is the same as for CC over λP2, by using Gödel's incompleteness theorem. The proof of conservativity of any system over a subsystem in the lower plane is again by normalization (of the proof terms).
The proof of conservativity of the right plane over the left plane can be done by defining a mapping that forgets predicates, analogously to the one defined in Definition 3.4. A slightly shorter proof can be given by making use of the conservativities in the Barendregt's cube, as follows: Let L_1 be a system in the left plane and let Γ be a context and A be a term such that $\Gamma \vdash_{L_1} A : \mathsf{Prop}$. Furthermore, let L_2 be the adjacent system in the right plane and let M be an inhabitant of A in L_2, that is $\Gamma \vdash_{L_2} M : A$. If S_1 is the system in Barendregt's cube that corresponds with L_1 and S_2 is the system in Barendregt's cube that corresponds with L_2, then $H(\Gamma) \vdash_{S_2} H(M) : H(A)$ and hence $\exists N[H(\Gamma) \vdash_{S_1} N : H(A)]$ by the conservativity in Barendregt's cube. Because the formulas-as-types embedding H is an isomorphism on the left plane of the cube, we can conclude that $\exists N[\Gamma \vdash_{L_1} N : A]$.
The proof of conservativity of $\lambda\text{PROP}\omega$ over λPROP2 is given in the following section. □

4. The conservativity of $\lambda\omega$ over $\lambda 2$

The conservativity of $\lambda\omega$ over $\lambda 2$ is shown by proving that PROPω is conservative over PROP2. The conservativity of $\lambda\omega$ over $\lambda 2$ then follows from the fact that the formulas-as-types embedding is an isomorphism. In order to be very specific about the conservativity proof, we first give the detailed syntax of the systems of second and higher order propositional logic.

4.1. Second and higher order propositional logics

4.1. DEFINITION. For n a natural number, the system of nth order propositional logic, notation PROPn is defined by first giving the nth order language and then describing the deduction rules for the nth order system as follows.

1. The *domains* are given by

$$\mathcal{D} ::= \mathsf{Prop} \mid (\mathcal{D} \to \mathcal{D}).$$

We let brackets associate to the right, so Prop→(Prop→Prop) is denoted by Prop→Prop→Prop and every domain can be written as $D_1 \to \ldots \to D_p \to \mathsf{Prop}$, with $D_1, \ldots, D_p$ domains.

2. The *order of a domain* D, $\mathsf{ord}(D)$, is defined by

$$\mathsf{ord}(\mathsf{Prop}) = 2,$$
$$\mathsf{ord}(D_1 \to \ldots \to D_p \to \mathsf{Prop}) = max\{\mathsf{ord}(D_i) \mid 1 \leq i \leq p\} + 1.$$

The orders are defined in such a way that in n-th order logic one can quantify over domains of order $\leq n$. Hence Prop is of order 2, because in second order propositional logic one can quantify over the set of all formulas. The domain Prop→Prop should be understood as the collection of sets of formulas (truth values), identifying a function P from Prop to Prop with the set of formulas φ for which $P\varphi$ holds.

3. For n a fixed positive natural number, the terms of the nth order language are defined as follows. (Each term is an element of a specific domain, which relation is denoted by ϵ).
   - There are countably many variables of domain D for any D with $\mathsf{ord}(D) \leq n$,
   - If $M \epsilon D_2$, x a variable of domain D_1 and $\mathsf{ord}(D_1 \to D_2) \leq n$, then $\lambda x \epsilon D_1 . M \epsilon D_1 \to D_2$,
   - If $M \epsilon D_1 \to D_2$, $N \epsilon D_1$, then $MN \epsilon D_2$,
   - If $\varphi \epsilon$ Prop, x a variable of domain D with $\mathsf{ord}(D) \leq n$, then $\forall x \epsilon D . \varphi \epsilon$ Prop.
   - If $\varphi \epsilon$ Prop and $\psi \epsilon$ Prop, then $\varphi \supset \psi \epsilon$ Prop.
4. The terms φ for which $\varphi \epsilon$ Prop are called *formulas* and Form denotes the set of formulas.
5. On the terms we have the well-known notion of definitional equality by β-conversion. This equality is denoted by $=$. The definitional equality allows us to identify for example the application of the function λx:Prop.$x \supset x$ (of domain Prop→Prop) to φ with the a formula $\varphi \supset \varphi$.
6. For n a specific positive natural number, we now describe the deduction rules of the nth order predicate logic (in natural deduction style) that allow us to build derivations. So in the following let φ and ψ be formulas of the nth order language.

$$(\supset\text{-I}) \quad \frac{\begin{array}{c}[\varphi]^i \\ \vdots \\ \psi\end{array}}{\varphi \supset \psi^{\,i}} \qquad (\supset\text{-E}) \quad \frac{\varphi \supset \psi \quad \varphi}{\psi}$$

$$(\forall\text{-I}) \quad \frac{\psi}{\forall x \epsilon D . \psi}(*) \qquad (\forall\text{-E}) \quad \frac{\forall x \epsilon D . \psi}{\psi[t/x]} \text{ if } t \epsilon D$$

$$(\text{conv}) \quad \frac{\psi}{\varphi} \text{ if } \varphi = \psi$$

The formula occurrences that are between brackets ([−]) in the $\supset$-I rule are *discharged*. So, in an application of the $\supset$-I rule, several occurrences of ϕ (possibly zero) may be discharged. The superscript i in the $\supset$-I rule is taken from a countable set of indices I. The index i corresponds to one specific application of the $\supset$-I rule, so from the index that is on top of a discharged formula, we can see at which application of $\supset$-I it has been discharged.
$(*)$: in the $\forall$-I rule we make the usual restriction that the variable x may not occur free in a non-discharged assumption of the derivation.
For Γ a set of formulas of PROPn and φ a formula of PROPn, we say that φ *is derivable from* Γ *in* PROPn, notation $\Gamma \vdash_{\mathrm{PROP}n} \varphi$, if there is a derivation with root φ and all non-discharged formulas in Γ.

The system of higher order proposition logic, notation PROPω, is the union of all PROPn.

4.2. REMARK. The choice for the connectives $\supset$ and $\forall$ may seem minimal. It is however a well-known fact that in second and higher order systems, the intuitionistic connectives $\&$, $\vee$, $\neg$ and $\exists$ can be defined in terms of $\supset$ and $\forall$ as follows. (Let φ and ψ be formulas).

$$\begin{aligned}
\varphi \,\&\, \psi &:= \forall \alpha \epsilon \mathsf{Prop}.(\varphi \supset \psi \supset \alpha) \supset \alpha,\\
\varphi \vee \psi &:= \forall \alpha \epsilon \mathsf{Prop}.(\varphi \supset \alpha) \supset (\psi \supset \alpha) \supset \alpha,\\
\bot &:= \forall \alpha \epsilon \mathsf{Prop}.\alpha,\\
\neg\varphi &:= \varphi \supset \bot,\\
\exists x \,\epsilon\, D\varphi &:= \forall \alpha \epsilon \mathsf{Prop}.(\forall x \epsilon D.\varphi \supset \alpha) \supset \alpha.
\end{aligned}$$

Similarly we can define an equality judgement (the β-equality $=$, the *definitional equality of the language*, is purely syntactical) by taking the so called Leibniz equality: for $t, q \,\epsilon\, D$,

$$t =_D q := \forall P \epsilon D \to \mathsf{Prop}.Pt \supset Pq,$$

which says that two objects are equal if they have the same properties. (It is not difficult to show that $=_D$ is symmetric).

It is not difficult to check that all the standard logical rules hold for $\&$, $\vee$, $\bot$, $\neg$, $\exists$ and $=$. In the following we shall freely use these symbols.

4.3. REMARK. In each PROPn ($n \geq 2$), the *comprehension* property is satisfied. That is, for all $\varphi(\mathbf{x})$: Prop with $\mathbf{x} = x_1, \ldots, x_p$ a sequence of free variables, possibly occurring in φ ($x_i \,\epsilon\, D_i$), we have

$$\exists P \,\epsilon\, D_1 \to \cdots D_p \to \mathsf{Prop}.\forall \mathbf{x} \epsilon \mathbf{D}(\varphi \leftrightarrow Px_1 \cdots x_p).$$

(Take $P \equiv \lambda x_1 \,\epsilon\, D_1 \ldots . \lambda x_p \,\epsilon\, D_p.\varphi(\mathbf{x})$.)

The presentation of propositional logics above can be extended to predicate logics, which is done in [Geuvers 1993]. There also the classical variants of the systems are studied. Here we restrict to the constructive versions of the propositional logics, because PROPω and PROP2 are the ones that correspond to the type systems $\lambda\omega$ and $\lambda 2$.

4.2. Extensionality

The definitional equality on the terms is β-equality. There is no objection to taking $\beta\eta$-equality instead: all the properties still hold. In fact it would make a lot of sense to do so, because we tend to view λ-abstraction as the necessary mechanism to make comprehension work. (And so both $P \epsilon \mathsf{Prop}\to\mathsf{Prop}$ and $\lambda x \epsilon \mathsf{Prop}.Px$ describe the collection of formulas φ for which $P\varphi$ holds).

This is related to the issue of *extensionality*: terms of domain $D\to\mathsf{Prop}$ are to be understood as predicates on D or also as subsets of D, an element t being in the set $P \epsilon D\to\mathsf{Prop}$ if Pt holds. But if we take this set-theoretic understanding seriously, we have to identify predicates that are extensionally equal:

$$(\forall \mathbf{x}.(f\mathbf{x} \supset g\mathbf{x} \;\&\; g\mathbf{x} \supset f\mathbf{x})) \supset f =_D g. \quad (1)$$

Of course, this formula is in general not provable in our systems. However, in the standard models where predicates are interpreted as real sets, the formula is satisfied, so it is an important extension. A difficulty is, that extensionality in the form of (1) is in general not even expressible: if $\mathsf{ord}(D) = n$ in PROPn, then we can not express extensionality for f and g of domain D, because $f =_D g$ is not a formula of PROPn (it uses a quantification over $D\to\mathsf{Prop}$). This means that we have to express extensionality by a schematic rule.

4.4. DEFINITION. The *extensionality scheme*, (EXT), is

$$(\mathrm{EXT})\ \frac{f\mathbf{x} \supset g\mathbf{x} \quad g\mathbf{x} \supset f\mathbf{x} \quad \varphi[f/y]}{\varphi[g/y]}\ (*)$$

where f and g are arbitrary terms of the same domain $D_1\to\cdots\to D_n\to\mathsf{Prop}$. When writing $\varphi[f/y]$, we assume that this substitution is correct, that is, no free variables become bound and f and y are of the same domain. $(*)$ signifies the usual restriction that the variables of $\mathbf{x}$ may not occur free in a non-discharged assumption of the derivations of $f\mathbf{x} \supset g\mathbf{x}$ and of $g\mathbf{x} \supset f\mathbf{x}$.
The extension of a system with the rule (EXT) will be denoted by adding the prefix E-, so E-PROPn is extensional nth order propositional logic.

4.5. NOTATION. For $f, g \epsilon D = D_1\to\cdots\to D_n\to\mathsf{Prop}$, if it is allowed to quantify over $D_1, \ldots, D_n$ in the system, then we can compress the first two premises in the rule (EXT) to $\forall \mathbf{x}.(f\mathbf{x} \supset g\mathbf{x} \;\&\; g\mathbf{x} \supset f\mathbf{x})$. For convenience this will also be denoted by $f \sim_D g$, so

$$f \sim_D g := \forall \mathbf{x}.(f\mathbf{x} \supset g\mathbf{x} \;\&\; g\mathbf{x} \supset f\mathbf{x}),$$

where the D will usually be omitted if it is clear from the context.

4.6. LEMMA. The extensionality scheme for $D = \mathsf{Prop}$ is admissible in any of the propositional logics, i.e.

$$\varphi \supset \psi, \psi \supset \varphi, \chi[\varphi/\alpha] \vdash \chi[\psi/\alpha]$$

is always provable.

PROOF. By an easy induction on the structure of χ. □

The following is now immediate by the fact that in PROP2 the only extensionality scheme that can be expressed is the one for $D = \mathsf{Prop}$.

4.7. COROLLARY. In the system E-PROP2 of extensional second order propositional logic one can prove the same as in PROP2. That is

$$\Gamma \vdash_{\text{E-PROP2}} \varphi \Leftrightarrow \Gamma \vdash_{\text{PROP2}} \varphi.$$

4.3. Algebraic semantics for intuitionistic propositional logics

In this section we describe a semantics for our systems of intuitionistic propositional logic in terms of Heyting algebras. It is well-known how this is done for the full first order propositional logic, giving rise to a completeness result. For second and higher order propositional logic we need to refine the notion of Heyting algebra to also allow interpretations for the universal quantifier. It will be shown that *complete* Heyting algebras are strong enough to satisfy our purpose: complete Heyting algebras have arbitrary meets and joins, so for example $\forall f \in \mathsf{Prop}\to\mathsf{Prop}.\varphi$ can be interpreted as $\bigwedge\{[\![\varphi]\!]_{[f:=F]} \mid F \in A\to A\}$. It is however not so easy to show the completeness of complete Heyting algebras over E-PROPn (for any n), because the Lindenbaum algebra defined from E-PROPn is not a complete Heyting algebra.

The proof of completeness that is given below uses Theorem 13.6.13 of [Troelstra and Van Dalen 1988], which states that any Heyting algebra can be embedded in a complete Heyting algebra such that $\supset$, $\bot$ and all existing $\bigvee$ and $\bigwedge$ are preserved (and hence the ordering is preserved). The embedding i that is constructed in the proof is also faithful with respect to the ordering, that is, if $i(a) \leq i(b)$ in the image, then $a \leq b$ in the original Heyting algebra. All this implies completeness of complete Heyting algebras with respect to E-PROPn, for any n. This will be shown in detail in the rest of this section. Hence we have conservativity of E-PROP$(n+1)$ over E-PROPn.

At this point we do not know how (if at all possible) to conclude the conservativity of PROP$(n+1)$ over PROPn from the conservativity of E-PROP$(n+1)$ over E-PROPn. However, we do have the conservativity of PROPn over PROP2 for any n, because PROP2 and E-PROP2 are the same system (Corollary 4.7).

It is obvious that extensionality is required in the syntax because the model notion is extensional: if, for example, $F, G : A\to A$ (where A is the carrier set of the algebra) and $F(a) = G(a)$ for all $a \in A$, then $F = G$.

The method of showing conservativity by semantical means seems to be quite essential here. Syntactic conservativity proofs (like the others in this paper) use

mappings from the 'larger' system to the 'smaller' system that are the identity on the smaller system. Such a mapping also constitutes a mapping from derivations to derivations, that is the identity on derivations of the smaller system. This was the case for the proof of conservativity of the upper plane of the cube over the lower plane, where the proof-term in the smaller system is just obtained by normalizing the proof-term in the larger system. For the case of propositional logics, this method is impossible: there are formulas of PROP2 that have more and more cut-free derivations when we go higher in the hierarchy of propositional logics. In Section 5 these issues will be discussed in some more detail.

4.8. DEFINITION. A *Heyting algebra* (or just Ha) is a tuple $(A, \wedge, \vee, \bot, \supset)$ such that $(A, \wedge, \vee)$ is a lattice with least element $\bot$ and $\supset$ is a binary operation with

$$a \wedge b \leq c \Leftrightarrow a \leq b \supset c,$$

where the ordering $\leq$ is defined by $a \leq b := a \wedge b = a$.

Remember that $(A, \wedge, \vee)$ is a lattice if the binary operations $\wedge$ and $\vee$ satisfy the following requirements.

$$\begin{array}{rr} a \wedge a = a, & a \vee a = a, \\ a \wedge b = b \wedge a, & a \vee b = b \vee a, \\ a \wedge (b \wedge c) = (a \wedge b) \wedge c, & a \vee (b \vee c) = (a \vee b) \vee c, \\ a \vee (a \wedge b) = a, & a \wedge (a \vee b) = a. \end{array}$$

Another way of defining the notion of lattice is by saying that it is a poset $(A, \leq)$ with the property that each pair of elements $a, b \in A$ has a least upperbound (denoted by $a \vee b$) and a greatest lowerbound (denoted by $a \wedge b$).

4.9. DEFINITION. A *complete Heyting algebra* (cHa) is a tuple $(A, \bigwedge, \bigvee, \bot, \supset)$ such that $(A, \bigwedge, \bigvee)$ is a complete lattice and $(A, \wedge, \vee, \bot, \supset)$ is a Heyting algebra. So $\bigvee$ and $\bigwedge$ are mappings from $\wp(A)$ to A such that for $X \subset A$, $\bigvee X$ is the least upperbound of X and $\bigwedge X$ is the greatest lower bound of X. The binary operations $\wedge$ and $\vee$ are defined by (for $a, b \in A$) $a \wedge b := \bigwedge\{a, b\}$ and $a \vee b := \bigvee\{a, b\}$.

An important feature of Heyting algebras which is forced by the presence of the binary operation $\supset$, is that they satisfy the infinitary distributive law:

$$(D) \quad \text{If } \bigvee X \text{ exists, then } a \wedge \bigvee X = \bigvee\{a \wedge b \mid b \in X\}.$$

The inclusion $\supseteq$ holds in any lattice; for the inclusion $\subseteq$ it is enough to show that $a \wedge c \subseteq \bigvee\{a \wedge b \mid b \in X\}$ for any $c \in X$, which is true due to the properties of $\supset$. Two other important facts are the following.

4.10. FACT. 1. If a complete lattice satisfies the infinitary distributive law (D), it can be turned into a cHa by defining

$$b \supset c := \bigvee\{d \mid d \wedge b \leq c\}.$$

2. Any Heyting algebra is distributive, i.e. any Ha satisfies

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c).$$

For the first statement one has to show that $a \wedge b \leq c \Leftrightarrow a \leq \bigvee\{d \mid d \wedge b \leq c\}$. From left to right is easy; from right to left, notice that if $a \leq \bigvee\{d \mid d \wedge b \leq c\}$, then $a \wedge b \leq b \wedge \bigvee\{d \mid d \wedge b \leq c\}$ and the latter is (by D) equal to $\bigvee\{b \wedge d \mid d \wedge b \leq c\}$, which is just c. The second is easily verified.

We are now ready to give the algebraic semantics for the systems E-PROPn. In the following let $(A, \bigwedge, \bigvee, \bot, \supset)$ be a cHa. We freely use the notions $\vee$ and $\wedge$, as they were given in Definition 4.9. The interpretation of the terms of E-PROPn will be in A and its higher order function spaces. We therefore let $\lceil - \rceil$ be the mapping that associates the right function space to a domain D, so

$$\lceil \mathsf{Prop} \rceil = A,$$
$$\lceil D_1 \rightarrow D_2 \rceil = \lceil D_1 \rceil \rightarrow \lceil D_2 \rceil,$$

where the second $\rightarrow$ describes function space. In the following we shall freely speak of the 'interpretation of E-PROPn in $(A, \bigwedge, \bigvee, \bot, \supset)$', where of course this interpretation includes the mapping of higher order terms into the appropriate higher order function spaces based on A.

Note that, as there are no constants in the formal systems of propositional logics, a model of E-PROPn is just a cHa: we do not need a valuation for the constants. (The extension with constants is no problem though. In that case a model is a pair $(\Theta, \mathcal{C})$, with Θ a cHa and $\mathcal{C}$ a mapping that assigns values to the constants.)

4.11. DEFINITION. The interpretation of E-PROPn in the cHa $(A, \bigwedge, \bigvee, \bot, \supset)$, $[\![-]\!]$, is defined modulo a valuation ρ for free variables that maps variables of domain D into $\lceil D \rceil$. So let ρ be a valuation. Then $[\![-]\!]_\rho$ is defined inductively as follows.

$$[\![\alpha]\!]_\rho = \rho(\alpha), \text{ for } \alpha \text{ a variable},$$
$$[\![PQ]\!]_\rho = [\![P]\!]_\rho([\![Q]\!]_\rho),$$
$$[\![\lambda x \epsilon D.Q]\!]_\rho = \boldsymbol{\lambda} t \in \lceil D \rceil . [\![Q]\!]_{\rho(x:=t)},$$
$$[\![\varphi \supset \psi]\!]_\rho = [\![\varphi]\!]_\rho \supset [\![\psi]\!]_\rho,$$
$$[\![\forall x \epsilon D.\varphi]\!]_\rho = \bigwedge\{[\![\varphi]\!]_{\rho(x:=t)} \mid t \in \lceil D \rceil\}.$$

Note that $[\![P]\!]_\rho([\![Q]\!]_\rho)$ denotes a set-theoretic function application and $\boldsymbol{\lambda}$ denotes set-theoretic abstraction.

It is easily seen that $[\![-]\!]_\rho$ satisfies the usual substitution property and that interpretations are stable under $\beta\eta$-equality, i.e.

$$[\![P]\!]_{\rho(x:=[\![Q]\!]_\rho)} = [\![P[Q/x]]\!]_\rho$$

and

$$P =_{\beta\eta} Q \Rightarrow [\![P]\!]_\rho = [\![Q]\!]_\rho.$$

4.12. DEFINITION. For Γ a set of formulas of E-PROPn, φ a formula of E-PROPn and Θ a cHa, φ *is* Θ*-valid in* Γ, notation $\Gamma \models_\Theta \varphi$, if for all valuations ρ,

$$\bigwedge\{[\![\psi]\!]_\rho \mid \psi \in \Gamma\} \leq [\![\varphi]\!]_\rho.$$

If Γ is empty we say that φ *is* Θ*-valid* if $\models_\Theta \varphi$.

In the following we just write $[\![\Gamma]\!]_\rho$ for $\bigwedge\{[\![\psi]\!]_\rho \mid \psi \in \Gamma\}$.

Our definition is a bit different from the one in [Troelstra and Van Dalen 1988], where $\Gamma \models_\Theta \varphi$ is defined by

$$\forall\rho((\forall\psi \in \Gamma[\![\psi]\!]_\rho = \top]) \Rightarrow [\![\varphi]\!]_\rho = \top).$$

Our notion implies the one above, but not the other way around. However, they are the same if $\Gamma = \emptyset$ and they also yield the same *consequence relation*.

4.13. DEFINITION. Let Γ be a set of formulas of E-PROPn and φ a formula of E-PROPn. We say that φ *is a consequence of* Γ, notation $\Gamma \models \varphi$, if $\Gamma \models_\Theta \varphi$ for all cHas Θ.

4.14. PROPOSITION (Soundness). For Γ a set of formulas of E-PROPn and φ a formula of E-PROPn,

$$\Gamma \vdash_{\text{E-PROP}n} \varphi \Rightarrow \Gamma \models \varphi.$$

PROOF. Let Θ be a cHa. By induction on the derivation of $\Gamma \vdash \varphi$ we show that for all valuations ρ, $[\![\Gamma]\!]_\rho \leq [\![\varphi]\!]_\rho$. None of the six cases is difficult. We treat the cases for the last rule being ($\supset$ -E) and ($\forall$-I).

($\supset$ -E) Say φ has been derived from $\psi \supset \varphi$ and ψ. Let ρ be a valuation. Then by IH $[\![\Gamma]\!]_\rho \leq [\![\psi]\!]_\rho$ and $[\![\Gamma]\!]_\rho \leq [\![\psi \supset \varphi]\!]_\rho$. The second implies $[\![\Gamma]\!]_\rho \wedge [\![\psi]\!]_\rho \leq [\![\varphi]\!]_\rho$. So, by $[\![\Gamma]\!]_\rho \leq [\![\psi]\!]_\rho$ we conclude $[\![\Gamma]\!]_\rho \leq [\![\varphi]\!]_\rho$.

($\forall$-I) Say $\varphi \equiv \forall f \epsilon D.\psi$ and $\Gamma' \subseteq \Gamma$ is the finite set of non-discharged formulas of the derivation with conclusion ψ. Then by IH, $\forall\rho[[\![\Gamma']\!]_\rho \leq [\![\psi]\!]_\rho]$, so $\forall\rho\forall F \in \lceil D \rceil[[\![\Gamma']\!]_\rho \leq [\![\psi]\!]_{\rho(f:=F)}]$, because $f \notin \text{FV}(\Gamma')$. This immediately implies that $[\![\Gamma]\!]_\rho \leq [\![\forall f \epsilon D.\psi]\!]_\rho$. □

To show completeness we first construct the Lindenbaum algebra for E-PROPn. This is a Ha but not yet a cHa. The construction in [Troelstra and Van Dalen 1988] tells us how to turn it into a cHa which has all the desired properties.

4.15. DEFINITION. For $n \in \mathbb{N} \cup \{\omega\}$, we define the *Lindenbaum algebra for* E-PROPn, $\mathcal{L}_n$. First we define the equivalence relation $\sim$ on Form(E-PROPn) by

$$\varphi \sim \psi \ := \vdash_{\text{E-PROP}n} \varphi \supset \psi \ \& \ \psi \supset \varphi.$$

We denote the equivalence class of φ under $\sim$ by $[\varphi]$. $\mathcal{L}_n$ is now defined as the Ha $(A, \wedge, \vee, \bot, \supset)$ where

$$\begin{aligned} A &= (\text{Form}(\text{E-PROP}n))_{\sim}, \\ [\varphi] \wedge [\psi] &= [\varphi \,\&\, \psi], \\ [\varphi] \vee [\psi] &= [\varphi \vee \psi], \\ [\varphi] \supset [\psi] &= [\varphi \supset \psi], \\ [\bot] &= [\bot]. \end{aligned}$$

Note that the $\&$, $\vee$, $\supset$ and $\bot$ on the right of the $=$ are the logical connectives: $\supset$ is basic and the others were defined in Remark 4.2 by

$$\begin{aligned} \varphi \,\&\, \psi &:= \forall \alpha \epsilon \mathsf{Prop}.(\varphi \supset \psi \supset \alpha) \supset \alpha, \\ \varphi \vee \psi &:= \forall \alpha \epsilon \mathsf{Prop}.(\varphi \supset \alpha) \supset (\psi \supset \alpha) \supset \alpha, \\ \bot &:= \forall \alpha \epsilon \mathsf{Prop}.\alpha. \end{aligned}$$

Each $\mathcal{L}_n$ is obviously a Ha: $[\varphi] \leq [\psi]$ iff $\varphi \vdash_{\text{E-PROP}n} \psi$.

4.16. LEMMA. For Γ a finite set of sentences of E-PROPn and φ a sentence of E-PROPn,

$$\Gamma \vdash_{\text{E-PROP}n} \varphi \;\Leftrightarrow\; [\bigwedge \Gamma] \leq [\varphi] \text{ in } \mathcal{L}_n.$$

PROOF. Immediate by the construction of $\mathcal{L}_n$. □

4.17. THEOREM ([Troelstra and Van Dalen 1988]). Each Ha Θ can be embedded into a cHa $c\Theta$ such that $\wedge$, $\vee$, $\bot$, $\supset$ and existing $\bigwedge$ and $\bigvee$ are preserved and $\leq$ is reflected.

PROOF. Let $\Theta = (A, \wedge, \vee, \bot, \supset)$ be a Ha. A *complete ideal of* Θ, or just *c-ideal*, is a subset $I \subset A$ that satisfies the following properties.

1. $\bot \in I$,
2. I is *downward closed* (i.e. if $b \in I$ and $a \leq b$, then $a \in I$),
3. I is *closed under existing sups* (i.e. if $X \subset I$ and $\bigvee X$ exists, then $\bigvee X \in I$).

Now define $c\Theta$ to be the lattice of c-ideals, ordered by inclusion. Then $c\Theta$ is a complete lattice that satisfies the infinitary distributive law D, and hence $c\Theta$ is a cHa by defining

$$I \supset J := \bigvee \{K \mid K \wedge I \subset J\}.$$

To verify this note the following.

- $c\Theta$ has infs defined by $\bigwedge_{q \in Q} I_q = \bigcap_{q \in Q} I_q$.
- $c\Theta$ has sups defined by $\bigvee_{q \in Q} I_q = \{\bigvee X \mid X \subset \bigcup_{q \in Q} I_q, \bigvee X \text{ exists}\}$: the set $\{\bigvee X \mid X \subset \bigcup_{q \in Q} I_q, \bigvee X \text{ exists}\}$ is indeed a c-ideal and it is also the least c-ideal containing all I_q.
- $I \cap \bigvee_{q \in Q} I_q = \bigvee \{I \cap I_q \mid q \in Q\}$ and so D holds.

The embedding i from Θ to $c\Theta$ is now defined by

$$i(a) = \{x \in A \mid x \leq a\}.$$

The embedding preserves $\bot$, $\supset$ and all existing $\bigwedge$, $\bigvee$. For the preserving of $\bigvee$, let $X \subset A$ such that $\bigvee X$ exists in Θ. We have to show that $i(\bigvee X) = \bigvee_{x \in X} i(x)$, i.e. show that

$$\{y \in A \mid y \leq \bigvee X\} = \{\bigvee Y \mid Y \subset \bigcup_{x \in X} i(x), \bigvee Y \text{exists}\}.$$

For the inclusion from left to right, note that $X \subset \{y \in A \mid \exists x \in X[y \leq x]\}$ and so $X \subset \bigcup_{x \in X} i(x)$. This implies that $\bigvee X \in \{\bigvee Y \mid Y \subset \bigcup_{x \in X} i(x), \bigvee Y \text{exists}\}$ and so we are done because the latter is a c-ideal. For the inclusion from right to left, let $z = \bigvee Y_0$ with $Y_0 \subset \bigcup_{x \in X} i(x)$. Then $z \leq \bigvee X$ so we are done.

Finally, the embedding i reflects the ordering, i.e.

$$i(a) \subset i(b) \Rightarrow a \leq b. \square$$

4.18. COROLLARY (Completeness). For Γ a finite set of sentences of E-PROPn and φ a sentence of E-PROPn,

$$\Gamma \models \varphi \Rightarrow \Gamma \vdash_{\text{E-PROP}n} \varphi.$$

PROOF. Following the Theorem, we embed the Lindenbaum algebra of E-PROPn, $\mathcal{L}_n$, in $c\mathcal{L}_n$. This cHa $c\mathcal{L}_n$ is complete with respect to the logic. So, for Γ a finite set of sentences and φ a sentence of E-PROPn, we have

$$\Gamma \models \varphi \;\Rightarrow\; \Gamma \models_{c\mathcal{L}_n} \varphi \;\Rightarrow\; [\bigwedge \Gamma] \leq [\varphi] \text{ in } \mathcal{L}_n \;\Rightarrow\; \Gamma \vdash_{\text{E-PROP}n} \varphi. \square$$

4.19. COROLLARY (Conservativity). For any $n \geq 2$, E-PROP$(n+1)$ is conservative over E-PROPn, and hence E-PROPω is conservative over E-PROPn.

PROOF. For Γ a finite set of sentences and φ a sentence of E-PROPn,

$$\Gamma \vdash_{\text{E-PROP}(n+1)} \varphi \Rightarrow \Gamma \models \varphi \Rightarrow \Gamma \vdash_{\text{E-PROP}n} \varphi$$

by soundness and completeness of the cHas for any of the E-PROPn.

The conservativity of E-PROPω over E-PROPn is now immediate: any derivation in E-PROPω is a derivation in E-PROPm for some $m \in \mathbb{N}$. $\square$

4.20. COROLLARY. For any $n \in \mathbb{N} \cup \{\omega\}$, PROP$n$ is conservative over PROP2.

PROOF. By the fact that PROPn is a subsystem of E-PROPn and the fact that PROP2 and E-PROP2 are the same system. $\square$

The formulas-as-types embedding (from PROPω to $\lambda\omega$, respectively from PROP2 to $\lambda 2$) is an isomorphism, so we can immediately conclude the following.

4.21. THEOREM. The type system $\lambda\omega$ is conservative over $\lambda 2$, that is, for all $\lambda 2$-contexts Γ and $\lambda 2$-types σ we have

$$\Gamma \vdash_{\lambda\omega} M : \sigma \Rightarrow \exists N[\Gamma \vdash_{\lambda 2} N : \sigma].$$

5. Discussion and concluding remarks

5.1. Semantical versus syntactical proofs of conservativity

We have seen that a proof of conservativity between typed λ calculi can be helpful to prove conservativity between logics. An example is the proof of conservativity of λPRED2 over λPRED, which immediately implies the conservativity of second order predicate logic over minimal first order predicate logic. Due to the syntactic nature of the proof (which is done by normalizing the proof terms), it is convenient to use the typed λ calculus format for the conservativity proof. For the proof of conservativity of $\lambda\omega$ over $\lambda 2$, a semantical proof was used. At this point it is not clear to us how a purely syntactical proof can be given for this case. For example, the method of normalizing the proof terms does not work here because of the following.

5.1. FACT. There are Γ, A and M, with Γ a context of $\lambda 2$, A a type of $\lambda 2$ and M a term of $\lambda\omega$, such that

$$\Gamma \vdash_{\lambda\omega} M : A \text{ and } \Gamma \not\vdash_{\lambda 2} \mathrm{nf}(M) : A.$$

One example is found by taking Γ to be the empty context and A to be the type of functions from numerals to numerals, so $A \equiv N \to N$, where $N \equiv \Pi\alpha : \star.(\alpha \to \alpha) \to \alpha \to \alpha$. Then one can take for M a representation of a recursive function that is λ-definable in $\lambda\omega$, but not λ-definable in $\lambda 2$. (Such terms exist, due to [Girard 1972], where it is shown that in $\lambda\omega$ more recursive functions are λ-definable then in $\lambda 2$.) Then $\vdash_{\lambda\omega} M : N \to N$, but not $\vdash_{\lambda 2} \mathrm{nf}(M) : N \to N$, because the normal form of M λ-defines the same recursive function as M.

An easier counterexample is found by taking, for example, the context Γ to be $x : \Pi\alpha{:}\star.\Pi\beta{:}\star.\beta \to \beta$. Then $\Gamma \vdash_{\lambda\omega} x(\Pi P{:}\star \to \star.P\bot) : \Pi\beta{:}\star.\beta \to \beta$, which is a term in normal form and not typable in $\lambda 2$.

It would be interesting to see how a syntactic proof of the conservativity of $\lambda\omega$ over $\lambda 2$ could be given. This would give an algorithm that computes for every $\lambda\omega$-term M that has a $\lambda 2$-type A, a $\lambda 2$-term N that also has the type A.

For the proof of conservativity of $\lambda 2$ over $\lambda\to$ (and hence of conservativity of second order propositional logic over minimal first order propositional logic), we used normalization here. This is not really necessary. It is possible to give a semantical proof, using, for example, the algebraic semantics that we discussed for second and higher order propositional logic. It is also possible to give a syntactic proof by defining a mapping from propositions and proofs of second order propositional logic to first order propositional logic that preserves derivability. This is done in [Pitts 1992].

5.2. Decidability

One of the consequences of conservativity of PROPω over PROP2 is that the system PROPω is not decidable. This follows from the undecidability of PROP2, which was proved by [Löb 1976] and also by [Gabbay 1981]. Similarly, all the

logics PROPn and E-PROPn are undecidable (for $n \in \mathbb{N} \cup \omega$), because they are all conservative over PROP2. In [Löb 1976] this is proved by giving a sound and complete translation of first order predicate logic into PROP2. The conservativity of PROPω over PROP2 shows that this translation does not extend to a sound and complete embedding of higher order predicate logic into PROPω. (Otherwise the 'first order parts' in PROPω and PROP2 could be used to show non-conservativity, in the style of the non-conservativity proof of λPREDω over λPRED2.)

5.3. Conservativity of λP2 over λP

The conservativity of λP2 over λP, and similarly the conservativity of λPRED2 over λPRED, may look a bit strange at first. Why does Gödel's Second Incompleteness Theorem not apply here? (Showing that the consistency of HA, first order arithmetic, can not be proved in HA itself; it can be proved in HA2, hence the non-conservativity of λPRED2 over λPRED.) The answer is that the logic λPRED is too minimal to be able to represent enough arithmetic to apply Gödel's Theorem. Remember that λPRED only has the connectives $\rightarrow$ and $\forall$, and because we are in a first order system, the other connectives are not representable. Hence an axiom like $\forall x \in \mathbb{N}[x \neq 0 \Rightarrow \exists y \in \mathbb{N}[x = Sy]]$ is not representable. The system λP has the same weakness.

We conjecture here that λP2 is not conservative over $\lambda\mathrm{P}^{\perp}$, where $\lambda\mathrm{P}^{\perp}$ is λP extended with a type constant $\perp$, a term operator $c_{\perp}$ and the rules

$$(\perp)\ \frac{}{\Gamma \vdash \perp : \star} \qquad (\perp\text{-I})\ \frac{\Gamma \vdash A : \star}{\Gamma \vdash c_{\perp} A : A}.$$

Of course, $\lambda\mathrm{P}^{\perp}$ is not really a subsystem of λP2, so it would be more precise to say that the (canonical) embedding from $\lambda\mathrm{P}^{\perp}$ into λP2 that maps $\perp$ to $\Pi\alpha{:}\star.\alpha$ is complete. This terminology is not used here, because we think it is clear that this is meant by the conservativity.

The motivation for this conjecture is that in $\lambda\mathrm{P}^{\perp}$, we can represent first order arithmetic, by using a classical interpretation of the connectives. So, for $\Gamma \vdash A, B : \star$ (with possibly $x{:}C$ free in A), define $\neg A :\equiv A \rightarrow \perp$, $A \,\&\, B :\equiv \neg(A \rightarrow \neg B)$, $A \vee B :\equiv \neg A \rightarrow B$ and $\exists x : C.A :\equiv \neg(\Pi x{:}C.\neg A)$. Then, classical first order arithmetic can be represented in $\lambda\mathrm{P}^{\perp}$ (and similarly in λP2). The consistency of this system can be stated in $\lambda\mathrm{P}^{\perp}$, but it can only be proved in λP2. Hence the non-conservativity of λP2 over $\lambda\mathrm{P}^{\perp}$.

References

[Barendregt 1992] H.P. Barendregt, Typed lambda calculi. In *Handbook of Logic in Computer Science*, eds. Abramski et al., Oxford Univ. Press.

[Barendsen and Geuvers 1989] E. Barendsen and H. Geuvers, λP is conservative over first order predicate logic, Manuscript, Faculty of Mathematics and Computer Science, University of Nijmegen, Netherlands,

[Berardi 1988] S. Berardi, Towards a mathematical analysis of the Coquand-Huet calculus of constructions and the other systems in Barendregt's cube. Dept. Computer Science, Carnegie-Mellon University and Dipartimento Matematica, Universita di Torino, Italy.

[Berardi 1989] S. Berardi, Talk given at the 'Jumelage meeting on typed lambda calculus', Edinburgh, September 1989.

[Berardi 1990] S. Berardi, Type dependence and constructive mathematics, Ph.D. thesis, Universita di Torino, Italy.

[De Bruijn 1980] N.G. de Bruijn, A survey of the project Automath, In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, eds. J.P. Seldin, J.R. Hindley, Academic Press, New York, pp 580-606.

[Coquand 1985] Th. Coquand, Une théorie des constructions, Thèse de troisième cycle, Université Paris VII, France, January 1985.

[Coquand 1990] Th. Coquand, Metamathematical investigations of a calculus of constructions. In *Logic and Computer Science*, ed. P.G. Odifreddi, APIC series, vol. 31, Academic Press, pp 91-122.

[Coquand and Huet 1988] Th. Coquand and G. Huet, The calculus of constructions, *Information and Computation*, 76, pp 95-120.

[Coquand and Huet 1985] Th. Coquand and G. Huet, Constructions: a higher order proof system for mechanizing mathematics. *Proceedings of EUROCAL '85, Linz*, LNCS 203.

[Gabbay 1981] D.M. Gabbay, *Semantical investigations in Heyting's intuitionistic logic*, Synthese Library, vol 148, Reidel, Dordrecht.

[Geuvers 1989] J.H. Geuvers, Talk given at the 'Jumelage meeting on typed lambda calculus', Edinburgh, September 1989.

[Geuvers and Nederhof 1991] J.H. Geuvers and M.J. Nederhof, A modular proof of strong normalisation for the calculus of constructions. *Journal of Functional Programming*, vol 1 (2), pp 155-189.

[Geuvers 1993] J.H. Geuvers, Logics and Type systems, PhD. Thesis, University of Nijmegen, Netherlands.

[Girard 1972] J.-Y. Girard, Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur. Ph.D. thesis, Université Paris VII, France.

[Girard et al. 1989] J.-Y. Girard, Y. Lafont and P. Taylor, *Proofs and types*, Camb. Tracts in Theoretical Computer Science 7, Cambridge University Press.

[Harper et al. 1987] R. Harper, F. Honsell and G. Plotkin, A framework for defining logics. *Proceedings Second Symposium on Logic in Computer Science*, (Ithaca, N.Y.), IEEE, Washington DC, pp 194-204.

[Howard 1980] W.A. Howard, The formulas-as-types notion of construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, eds. J.P. Seldin, J.R. Hindley, Academic Press, New York, pp 479-490.

[Löb 1976] M. Löb, Embedding first order predicate logic in fragments of intuitionistic logic, *J. Symbolic Logic* vol 41, 4, pp. 705–719.

[Longo and Moggi 1988] G. Longo and E. Moggi, Constructive Natural Deduction and its "Modest" Interpretation. Report CMU-CS-88-131.

[Paulin 1989] Ch. Paulin-Mohring, Extraction des programmes dans le calcul des constructions, Thèse, Université Paris VII, France.

[Pitts 1992] A.M. Pitts, On an interpretation of second order quantification in first order intuitionistic propositional logic. *J. Symbolic Logic* vol 57, 1, pp.33–52.

[Ruys 1991] M. Ruys, $\lambda P\omega$ is not conservative over $\lambda P2$, Master's thesis, University of Nijmegen, Netherlands, November 1991.

[Tonino and Fujita 1992] H. Tonino and K.-E. Fujita, On the adequacy of representing higher order intuitionistic logic as a pure type system, *Annals of Pure and Applied Logic* vol 57, pp 251–276.

[Troelstra and Van Dalen 1988] A. Troelstra and D. van Dalen, *Constructivism in mathematics, an introduction, Volume I/II*, Studies in logic and the foundations of mathematics, vol 121 and volume 123, North-Holland.

[Verschuren 1990] E. Verschuren, Conservativity in Barendregt's cube, Master's thesis, University of Nijmegen, Netherlands, December 1990.